Lecture 06: Stable vs. Maximal Matchings

Overview

1. Recap of last time
   - Locality Lemma
   - SM lower bound

2. Maximal Matchings

3. Comparing Stable and Maximal Matchings

4. Looking Forward: Detecting Termination in the LOCAL Model

Last Time

- Proved the Locality Lemma

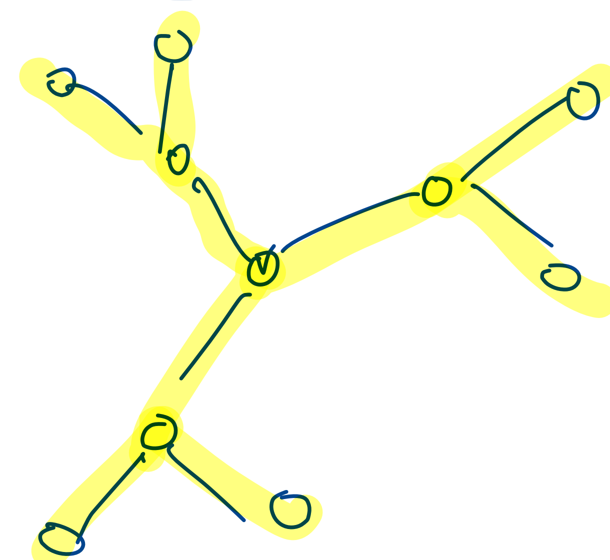
- used Locality Lemma to prove a lower bound for SMP:

$\boxed{\text{Any distributed protocol that computes stable matchings requires } D \text{ rounds on (for some preferences) on graphs w/ diameter } D.}$

- Computing stable matchings is a global Problem
  - SMs cannot be found w/ only local information

Locality Lemma

- Fix graph $G = (V, E)$ and protocol Π
- for any $v \in V$ and round r, v's state in round r is determined by $\Gamma_{r-1}(v)$



A More Algorithmic View of Locality Lemma

In round 1, consider $\Gamma_{r-1}(v)$

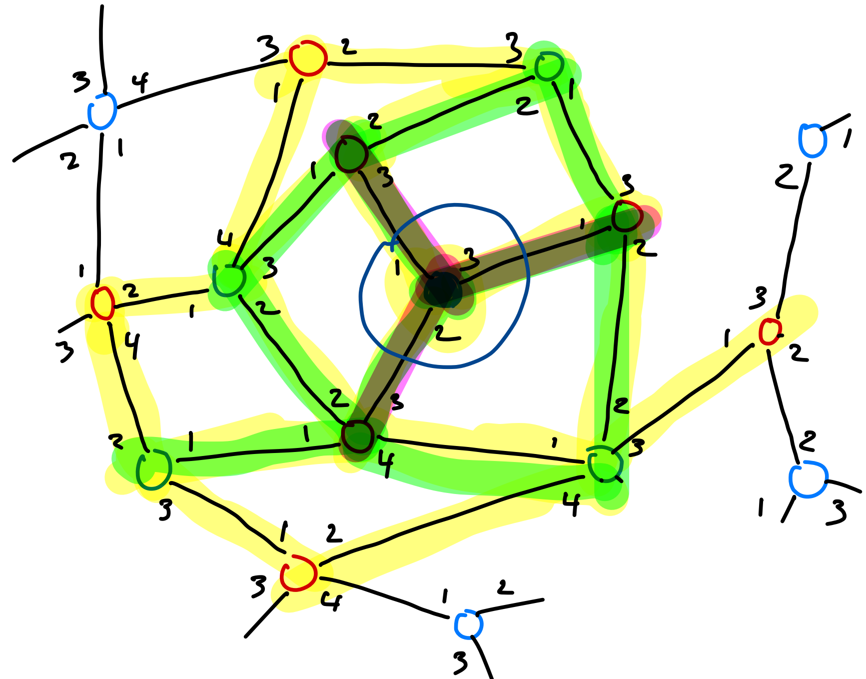
- compute all init. states
- compute all msgs
- determine all received msgs in $\Gamma_{r-2}(v)$

In round 2, consider $\Gamma_{r-2}(v)$

- compute all rnd 2 states
- compute all rnd 2 msgs
- find all received msgs in $\Gamma_{r-3}(v)$

⋮

In round $r-1$, get all states / received msgs in $\Gamma_0(v) = \{v\}$.

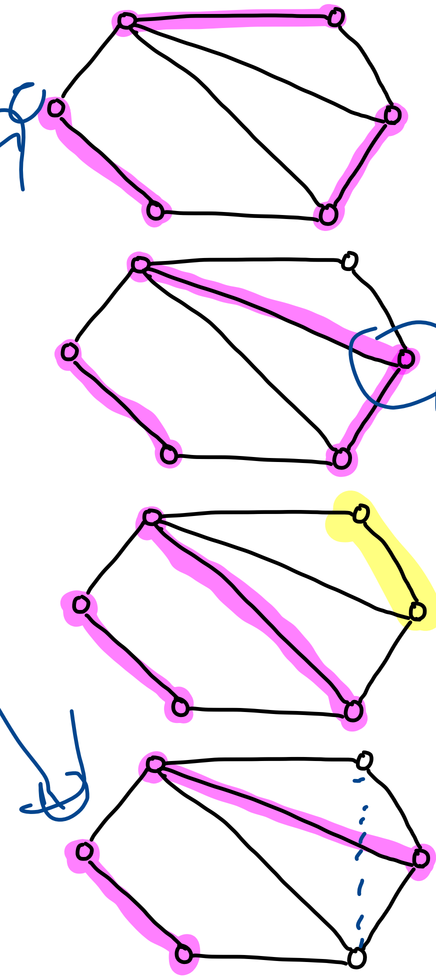


Maximal Matchings

- G = (V, E) a graph
- M ⊆ E (subset) a set of edges is a <u>matching</u> if each vertex is incident to at most one edge in M
- A matching M is a <u>maximal matching</u> if there is no larger matchim M' that contains M
  - Equivalently, M is maximal if every v ∈ V is either incident to an edge in M, or all of v's neighbors are incident to edges
  - informally: no edge can be added to M to result in a matching

[maximum (cardinality) matching = largest possible max'l

both max'l



Matching, Maximal, "perfect matching"

not matching

matching, not maximal

Maximal matching

Which are (maximal) matchings?

Exercise / Meditation:

Show that computing maximum cardinality matchings is a <u>global</u> problem

↳ needs $\gtrsim D$ rounds on graphs w/ diameter D.

Stable vs Maximal Matchings

- Consider context of SMP
  - PO network
  - 2 colored: blue nodes are students, red nodes are internships
- What if we are content to find a maximal matching between students and internships?

Big Question. Can maximal matchings be found efficiently?

Assume. No student applies to more than Δ internships

- maximum (student) degree is $\le \Delta$

Take $\Delta = O(1)$

- constant, independent of # nodes

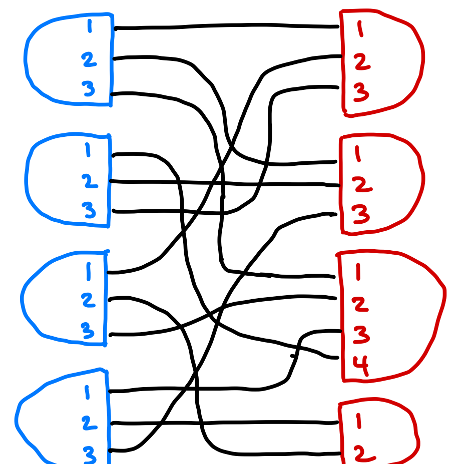
SM Instance

	1	2	3	4
Anna:	a	b	c	
Beck:	c	b	a	
Cameron:	a	d	c	
Daniel:	c	d	b	

	1	2	3	4
a:	A	C	B	
b:	A	B	D	
c:	A	C	D	B
d:	D	C		

$\Downarrow$

PO Network



"2-colored bipartite graphs"

A distributed algorithm for maximal matchings?

- Start w/ Gale-Shapley
- Does it work? ← yes b/c stable matchings are maximal
- Can it be made more efficient if we don't require stability?

Procedure for students:

```
Initialize:
  cur = 1
each round i, do
  if i = 1, send "apply" to cur
  if received "reject" from cur in round i-1
    if cur = my degree, return ⊥ and halt
    else, cur ← cur+1, send "apply" to cur
return cur and halt
```

Procedure for internships

```
Initialize:
  cur = ∞
each round i, do
  set reject ← ∅
  for each j from which received
  "apply" in round i-1
    if j < cur
      add cur to reject
      set cur ← j
    else
      add j to reject
  for each j in reject
    send "reject" to j
return cur
```

Union Rules

1. All applications require a response within 1 round of receipt
   - response = "accept" or "reject"
   - no option to defer
2. Once an offer is accepted, it cannot be revoked

How to modify Internship Procedure?

if receive any app, accept best in first rnd that I receive, reject all subsequent apps.

Procedure for internships

Initialize:

cur = ∞

each round i, do

```
    set reject ← ∅
    for each j from which received "apply" in round i-1
        if j < cur
            add cur to reject
            set cur ← j
        else
            add j to reject
    for each j in reject
        send "reject" to j
return cur
```

Students still apply only to one internship at a time

How to modify Student Procedure?

if receive "accept", then terminate

Procedure for students:

Initialize:

cur = 1

each round i, do

if $i = 1$, send "apply" to cur

if received "reject" from cur in round $i-1$

if cur = my degree, return $\perp$ and halt

else, cur $\leftarrow$ cur $+1$, send "apply" to cur

return cur and halt

Protocol for $\boxed{\text{Maximal Matchings?}}$

Student Procedure

Initialize:

cur = 1

Each round i:

```
if i = 1
    send cur "apply"
if received "accept"
    output cur and halt
if received "reject"
    if cur < my degree
        cur ← cur + 1
        send "apply" to cur
    else
        output ⊥ and halt
```

Internship Procedure

Initialize

cur = ⊥, matched = false

Each round i

```
if received "apply"
    if matched
        respond to all apps w/ "reject"
    else
        cur ← most favored app received
        matched ← true
        send cur "accept"
        send others "reject"
```

Does this Work?

Correctness of Maximal Matching Alg.

Termination. If every student applies to at most Δ internships, then every node student halts in at most $2\Delta+1$ rounds.

Why?

- Students send at most Δ apps.
- each app gets response in next round

1: apply
2: receive
3: apply (if not matched)
4: receive
⋮
2Δ: receive, $2\Delta+1$ halt

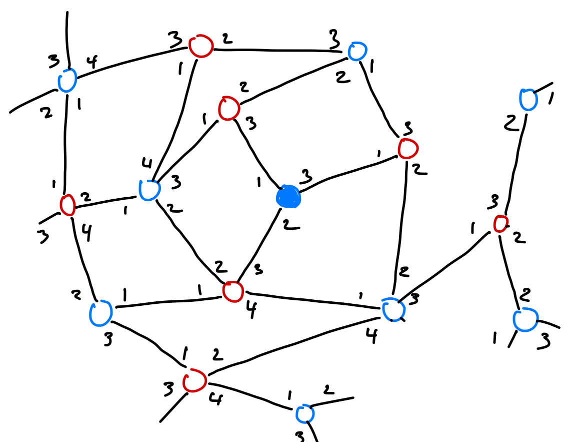
Maximality. Let M be the matching of accepted applications. Then M is a maximal matching.

Why?

If S is unmatched at end, rejected by all acceptable partners,
each one accepted an app before rejecting
so S unmatched
$\Rightarrow$ all neighbors are matched

Comparing Maximal vs Stable Matching

- Port-ordered network
- each node is a student or internship
- each student applies to at most Δ internships
  - $\Delta = O(1)$ (const)
  - network has bounded (student) degree
- total # of agents is n, total # of edges (applications) is m ($\leq n \cdot \Delta$)



Comparing Maximal vs Stable Matching

	Stable Matchings	Maximal Matchings
Centralized alg. running time	$O(n^2)$, $O(\Delta n)$, $O(\underline{n}+m)$, $\boxed{O(m)}$	$O(n+\underline{\underline{m}})$, $\boxed{O(m)}$
Distributed alg. running time (rounds)	$\rightarrow O(m)$ gap	$\rightarrow O(\Delta) \leftarrow$ does not depend on size of network
Lower Bound?	D rounds; could be up to n	$\underline{1}$

Note. Maximal matchings can also be found for general (not "2-colored") graphs in $O(\underline{\Delta} + \log^* n)$ rounds. There are matching lower bounds of $\underline{\Omega(\log^* n)}$ (Linial 1987) and $\underline{\Omega(\Delta)}$ (Balliu et al 2019).

The Moral

- In <u>centralized</u> computing, SM and MM are very similar
  - both admit elegant <u>linear time</u> algorithms

- In the <u>distributed</u> setting, they are very different
  - MM can be solved in $O(\Delta)$ rounds
    - for const. Δ, MM can be solved <u>locally</u>
    - locality lemma $\Longrightarrow$ seeing dist 2Δ neighborhood is sufficient to determine output
  - SM cannot be solved in less than D (= diameter) rounds

Lingering Question

Can we actually compute SMs in the PO model?

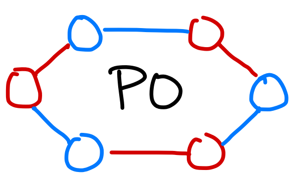
- GS algorithm works... except termination detection
- if all nodes know n (= # agents) or m (= # applications) then they can terminate after $2 \cdot m$ or $2 \cdot n \cdot \Delta$ (or $2 \cdot n^2$) rounds

What if m and n are unknown to nodes?

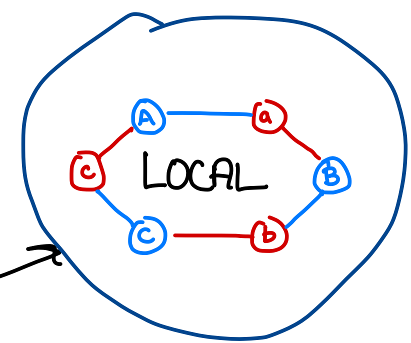
- can we detect termination in PO model?

- can we detect termination in LOCAL model?

  LOCAL = PO + unique IDs



how different are these scenarios?



Coming Up

Explore the role of communication in distributed algorithms

Locality = how far away is info I need to find my local output
= # of rounds w/ unrestricted communication

Communication = how much info do I need to find my local output

E.g. Gale-Shapley only sends 2 distinct messages
- potentially slow (global)
- uses little communication: $\Theta(m)$ bits

Next Goal. We can detect termination w/out too much overhead

- BFS tree construction
- "Leader Election"

} Use: unique IDs, small messages

"CONGEST" Model.